IMPLEMENTACIÓN EN CHIP VLSI DEL ALGORITMO CORDIC PARA LA SOLUCIÓN DE FUNCIONES TRIGONOMÉTRICAS

VLSI ON-CHIP IMPLEMENTATION OF THE CORDIC ALGORITHM FOR THE SOLUTION OF TRIGONOMETRIC FUNCTIONS

Ramón Chávez Bracamontes
Tecnológico Nacional de México en Ciudad Guzmán
rachavez@itcg.edu.mx

Víctor Manuel Vidrios Núñez
Tecnológico Nacional de México en Ciudad Guzmán
victorm15290023@itcg.edu.mx

José Refugio Villaseñor Salvatierra
Tecnológico Nacional de México en Ciudad Guzmán
joseref@msn.com

Humberto Bracamontes del Toro
Tecnológico Nacional de México en Ciudad Guzmán
hbdeltoro@gmail.com

Jesús Ezequiel Molinar Solís
Tecnológico Nacional de México en Ciudad Guzmán
molinar@gmail.com

Marco Antonio Gurrola Navarro
Universidad de Guadalajara–Centro Universitario de Ciencias Exactas e Ingeniería
marco.gurrola@cucei.udg.mx

Resumen

En este trabajo se presenta una implementación del algoritmo CORDIC en tecnología VLSI con una arquitectura de 16 bits. Se propone una forma simplificada para resolver las funciones trigonométricas básicas por hardware con tiempos de ejecución de entre 16 y 36 ciclos de reloj. El diseño de la arquitectura fue sintetizado para fabricarse en un proceso de tecnología CMOS de 0.5 micras de On Semiconductor sobre un área de 1.88 mm². Este diseño nos habilita la generación de módulos IP aritméticos de alta eficiencia que nos permitirá el

desarrollo de algoritmos en chips tipo ASIC indispensables en la línea de investigación de integración de sistemas digitales.

Palabra(s) Clave: Diseño VLSI, Funciones trigonométricas, Alliance CAD System, Algoritmo en Hardware.

Abstract

In this paper we present an implementation of the CORDIC algorithm in VLSI technology with a 16 bit architecture. A simplified form is proposed to solve the basic trigonometric functions by hardware with execution times of between 16 and 36 clock cycles. The design of the architecture was synthesized to be manufactured in a 0.5 micron CMOS technology process of On Semiconductor over an area of 1.88 mm². This design enables us to generate high efficiency arithmetic IP modules that will allow us to develop algorithms on ASIC-type chips that are indispensable in the digital systems integration research line.

Keywords**:** *VLSI Design, Square Root, Alliance CAD System, Algorithm on Chip.*

1. Introducción

El algoritmo CORDIC (COrdinate Rotation DIgital Computer) fue introducido por (Volder, 1959) para el cálculo de funciones trigonométricas, multiplicación, división y cambio de base entre sistemas, posteriormente en 1971 J.S Walther realizó una unificación del algoritmo para resolver más funciones como funciones hiperbólicas, raíz cuadrada, logaritmo y exponencial (Walter, 1971, 2000), desde entonces hasta la actualidad, CORDIC ha sido estudiado para una gran cantidad de aplicaciones en las cuales han surgido diferentes propuestas de mejora en los recursos utilizados (Pramod y Meher, 2009), (Chen y Wen, 2003), (Takagi et al., 1991), (Duprat y Muller, 1993), (Valls, 2000). El algoritmo CORDIC se evalúa de manera iterativa mediante la integración de funciones básicas de sumas, restas, multiplicaciones, y registros de desplazamiento entre otras unidades aritméticas funcionales. En este artículo se presenta una arquitectura de 16 bits de punto fijo, optimizada para la solución de funciones trigonométricas básicas como seno, coseno, tangente, secante y cosecante. Para la síntesis digital de la arquitectura

propuesta se hace uso de las herramientas de uso libre como Alliance CAD System (LIP6, 2018), estas herramientas de diseño asistido por computadora (CAD) permiten el uso de celdas estándar a nivel VLSI, que garantizan un máximo rendimiento en el área utilizada. Para validar la arquitectura se diseñaron los Layouts requeridos para el diseño de un chip en tecnología CMOS de 0.5 micras de On Semiconductor. La fabricación del chip se realizó mediante un convenio establecido entre el Instituto Tecnológico de Ciudad Guzmán y la compañía MOSIS - Metal Oxide Semiconductor Implementation Service (The MOSIS Service, USC Information Sciences Institute, 2018). Se realizaron las pruebas de verificación funcionales de los chips recibidos y finalmente se validaron los resultados experimentales contra los de simulación. Se experimentó con un rango de valores para identificar la magnitud del error en el cálculo de las funciones trigonométricas y se compararon los resultados contra los obtenidos por su implementación por software. El análisis de los errores por efectos de la precisión de 16 bits empleada arrojó errores relativos de ±0.034% alcanzados en solo 7 iteraciones mediante el método propuesto. Error que, comparado con el mínimo error posible de $1/2\text{^}16 = 0.002\%$ con datos de 16 bits, demuestra la funcionalidad de la arquitectura propuesta, misma que puede implementarse para logra una mayor exactitud en los resultados extendiendo la arquitectura a 24 o 32 bits en trabajos futuros.

2. Métodos

Algoritmo CORDIC

El algoritmo CORDIC (Muller, 2016) es un método computacional para calcular diferentes funciones mediante el uso de la rotación de ángulos como se muestra en la figura 1, este artículo se enfoca solo en la solución de funciones trigonométricas, para ello se parte de las ecuaciones 1, 2 y 3 (Bruce, 2000) y una constante de escalamiento según la ecuación 4 (Volder, 1959), donde x, y, z representan al coseno, seno y ángulo respectivamente, la constante m determina la función que realizará el algoritmo, que para funciones trigonométricas es igual a

uno, δk es el signo que depende de z, si es mayor o igual a cero entonces el valor es positivo de lo contrario el valor es negativo, σk son valores almacenados dependiendo de las iteraciones, una vez teniendo estos resultados se realiza una multiplicación por la constante $K \approx 0.6072$ para obtener el resultado final.

$$x_{k+1} = x_k - m\delta_k y_k 2^{-k} \quad (1)$$

$$y_{k+1} = y_k + \delta_k x_k 2^{-k} \quad (2)$$

$$z_{k+1} = z_k - \delta_k \sigma_k \quad (3)$$

$$K = \frac{1}{1.646760255} \quad (4)$$

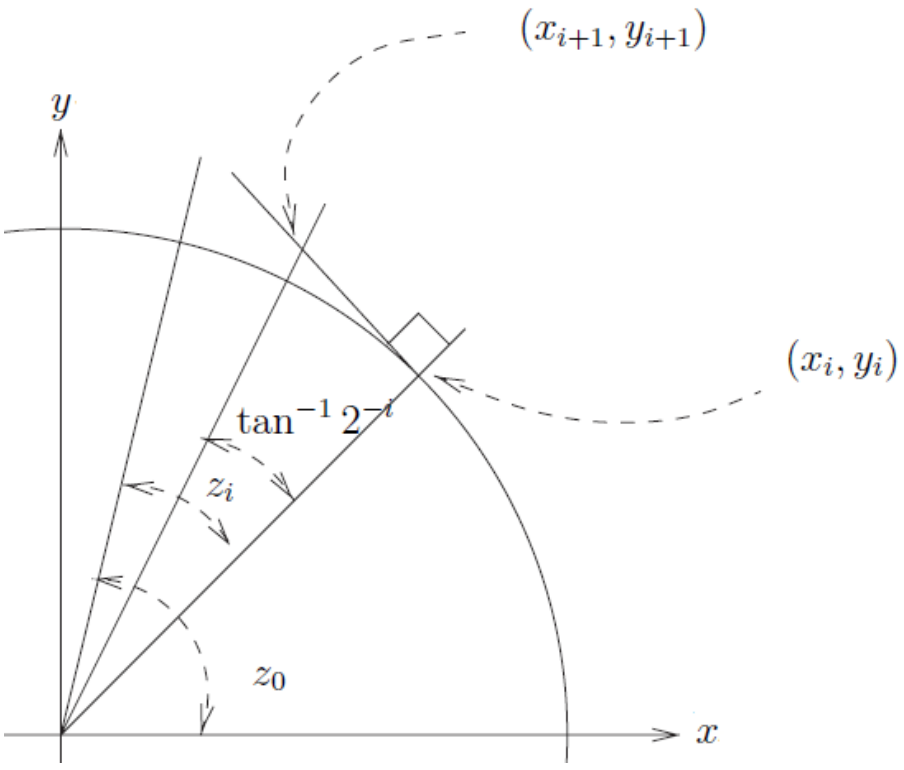

Figura 1 Iteración del algoritmo CORDIC.

La figura 2 muestra el diagrama flujo del algoritmo CORDIC representado por las ecuaciones 1, 2 y 3.

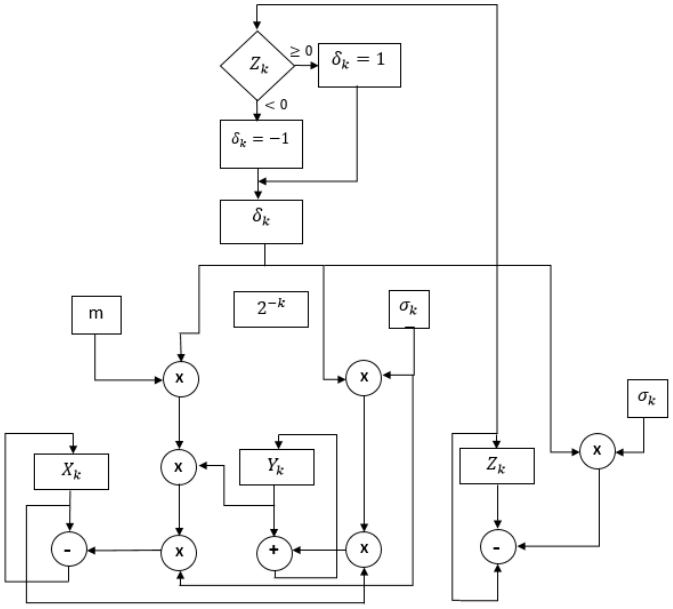

Figura 2 Diagrama de flujo del algoritmo CORDIC.

Arquitectura TLU

Para implementar en hardware el algoritmo CORDIC, se diseñó la arquitectura para un circuito integrado denominado Unidad Lógica Trigonométrica (TLU). Las funciones trigonométricas que la TLU resuelve son específicamente el seno, coseno, tangente y sus inversas como la secante, cosecante y cotangente de un ángulo de entrada. La solución de dichas funciones se realiza de forma iterativa, por lo cual es necesario indicar con una señal de inicio el momento para comenzar las iteraciones y por medio de otra señal de estado identificar la finalización de las iteraciones en el momento en que se tiene disponible el resultado. El chip TLU se diseñó para un encapsulado en formato DIP 40, de los cuales 23 pines son de entrada, 17 de salida y 2 de alimentación como se muestra en la figura 3. La función del TLU es simple, se ingresa un ángulo en formato binario de 16 bits donde el bit más significativo es el signo, 7 bits para la parte entera y 8 bits para la parte fraccionaria. Para seleccionar la función se emplean 3 pines de control denominados *ctl* (tabla 1). Se comienza verificando que esté presente la señal de reloj en la terminal *clock* y para comenzar el cálculo de la función seleccionada se aplica un pulso de un periodo de reloj al pin denominado *start*. La terminal de salida de *busy* estará activada durante el periodo de cálculo y cuando *busy* se

vaya al estado bajo estará disponible el resultado en las terminales de salida *dout*, las cuales se obtienen en formato binario de 16 bits, donde 12 bits corresponden a la parte fraccionaria, 3 bits para la parte entera y uno para el signo (formato Q12).

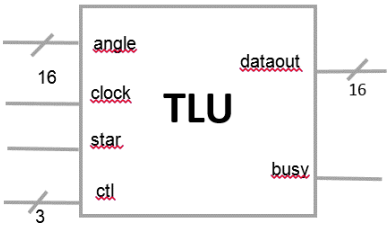

Figura 3 Unidad Lógica Trigonométrica (TLU).

Tabla 1 Selección de la función trigonométrica.

Función	Seno	Coseno	Tangente	Cosecante	Secante
ctl	000	001	010	011	100

La arquitectura propuesta se compone por los elementos principales mostrados en la figura 4, dentro de los que destacan los módulos CALTRI, FSM y RATIO. En el módulo CALTRI se llevan a cabo de forma iterativa las operaciones definidas en las ecuaciones 1, 2 y 3 del algoritmo CORDIC. FSM es la máquina de estados que controla los tiempos de ejecución de los módulos principales CALTRI y RATIO. El módulo RATIO tiene la capacidad de realizar operaciones de multiplicación, reciproco y división de datos de 16 bits provenientes del módulo CALTRI.

La arquitectura del módulo CALTRI se muestra en la figura 5 implementada a base de sumadores, restadores, multiplexores y registros de desplazamiento.

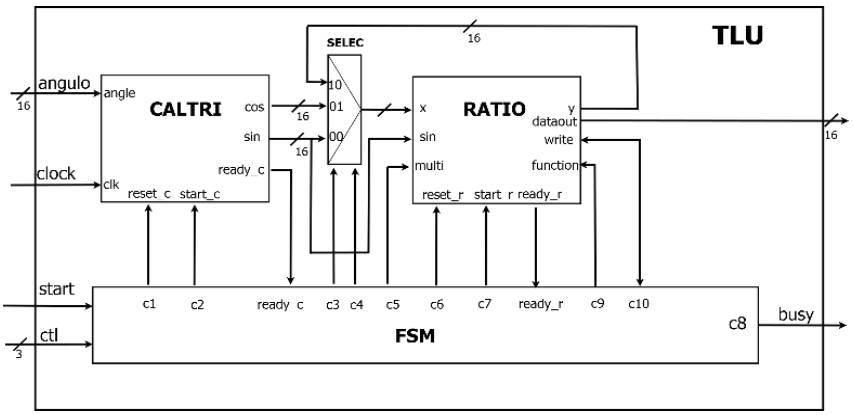

Figura 4 Arquitectura de TLU.

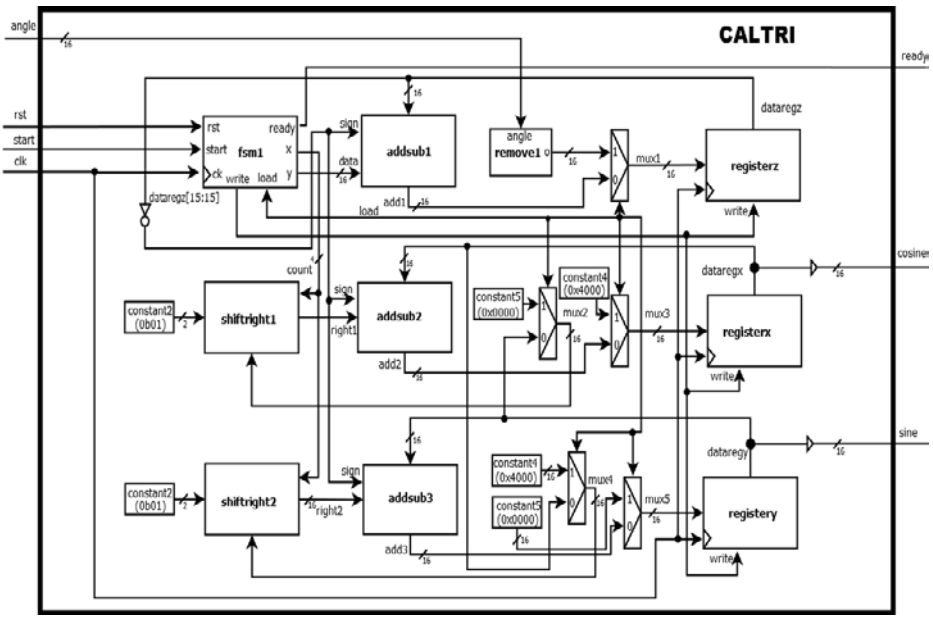

Figura 5 Arquitectura del módulo CALTRI.

En la figura 6 se muestra la arquitectura del módulo RATIO. Con el arreglo de multiplexores se seleccionan las funciones de seno, coseno, tangente, secante y cosecante.

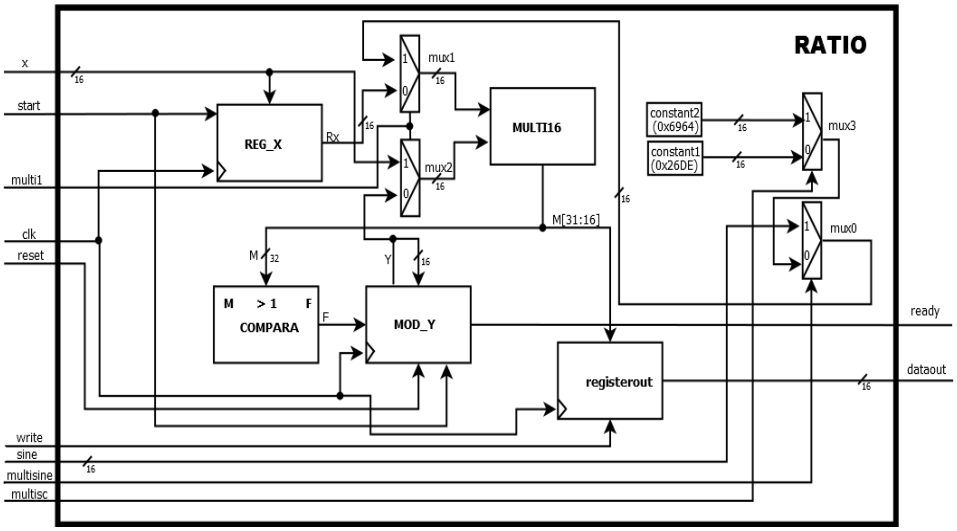

Figura 6 Arquitectura del módulo RATIO.

Para calcular el reciproco de un valor determinado se ingresa el valor por la terminal *x* en formato Q12, se aplica un pulso a la terminal de *start* para comenzar

y mediante aproximaciones sucesivas se realiza el cálculo de reciproco del valor dado, la señal en la terminal de *busy* indica cuando se está haciendo el cálculo; cuando la terminal *busy* se encuentra en el estado lógico ´1´, está realizando el cálculo mediante una serie de iteraciones y cuando se encuentra en el estado lógico ´0´, es una indicación que ha terminado el cálculo de la función en cuestión, y el valor puede ser leído por la terminal *dataout* en formato Q12. Para multiplicar el valor en la terminal *x* por un factor de escalamiento o en su caso por la función de seno se habilita las terminales de *multi1*, *multisc* y *multisine*.

Herramientas de síntesis digital: Alliance CAD System

Los circuitos integrados digitales han estado en constante mejora debido a los requerimientos de los diseñadores para realizar tareas cada vez más complejas, que como consecuencia permiten el desarrollo de esquemas optimizados que facilitan el proceso de diseño con una mínima cantidad de errores.

Alliance CAD System es una herramienta para el diseño digital VLSI (Weste y Harris, 2011), el cual proporciona un compilador para el lenguaje de descripción de hardware (VHDL) y las herramientas de síntesis que incluyen librerías de celdas estandarizadas facilitando el diseño digital. La metodología empleada para la síntesis VLSI está basada en (Chaput y Pétrot , 2002), (Lam y Ak, 2004), la cual comienza desde la descripción en VHDL para cada uno de los módulos definidos en la arquitectura propuesta. Posteriormente se hace uso de diferentes herramientas como *vasy*, *boom, boog, loon* que son utilizadas como herramientas de síntesis (Lam y Ak, 2004). Además, las herramientas de Place y Route como *ocp, nero* y *s2r* que permiten generar la descripción de los archivos con la descripción física del circuito integrado. Mediante la herramienta GENLIB (Lam y Ak, 2004) se define en forma estructural la arquitectura completa y para verificar el diseño del circuito integrado propuesto se utilizan las herramientas *asimut, cougar* y *lvx* (LIP6, 2018), las cuales nos permiten la detección de errores durante la simulación de cada uno de los módulos que conforman la arquitectura, así como la simulación del circuito completo. Una vez que se valida la arquitectura y se sintetizan los archivos que describen al circuito integrado y habiendo cumplido con

las reglas de diseño, entonces puede ser enviado a fabricar con un elevado grado de confiabilidad.

3. Resultados

Los resultados obtenidos en la implementación de la arquitectura se verificaron a través del uso de las herramientas de simulación digital de Alliance CAD System, para esto se preparó un archivo con patrones de datos que nos sirvieron para estimular el módulo TLU mediante la herramienta de simulación *asimut* (Lam y Ak, 2004).

Para validar la arquitectura propuesta se diseñaron los Layouts requeridos para la fabricación del chip TLU empleando tecnología CMOS de 0.5 micras de On Semiconductor. el cual se fabricó mediante un convenio de colaboración con la compañía MOSIS y el Instituto Tecnológico de Ciudad Guzmán.

Para la verificación de los resultados del chip fabricado se realizaron pruebas funcionales con 5 chips. Se experimentó con un rango de valores para identificar la magnitud del error en el cálculo de las funciones trigonométricas seno, coseno, secante, cosecante y tangente y se compararon los resultados contra los obtenidos por su implementación con el software de Excel. Se realizó el análisis de los errores por efectos de la precisión de 16 bits empleada, el cual arrojó errores relativos de ±0.034 % alcanzadas en solo 7 iteraciones del algoritmo propuesto. La tabla 2 muestra los resultados experimentales con el chip TLU y se comparan con los valores calculados mediante el algoritmo CORDIC implementado en Excel.

Tabla 2 Resultados del chip TLU vs. algoritmo CORDIC implementado en Excel.

Función	**Argumento**					
	46.25°		**25°**		**60°**	
	Chip TLU	**Excel**	**Chip TLU**	**Excel**	**Chip TLU**	**Excel**
cos	0.6953125	0.69541366	0.91015625	0.91017084	0.497070313	0.49717158
sen	0.71850586	0.71843173	0.41381836	0.41392433	0.867431641	0.86750488
sec	1.39160156	1.3919207	1.09838867	1.09869483	2.010986328	2.01137805
csc	1.43774414	1.43799304	2.41577148	2.41590053	1.15234375	1.15273126
tan	1.03309983	1.04461758	0.45466738	0.45477652	1.745088409	1.74488028

Estos valores presentan una gran precisión, lo que nos habla de una arquitectura implementada de manera eficiente obteniendo valores de errores relativos máximos del 0.034 % como se muestra en la tabla 3. Validando la factibilidad de la arquitectura VLSI propuesta.

Tabla 3 Errores relativos porcentuales del chip vs. calculados en Excel.

	Argumento		
Función	**46.25°**	**25°**	**60°**
cos	0.014	0.002	0.020
sen	0.010	0.026	0.008
sec	0.023	0.028	0.019
csc	0.017	0.005	0.034
tan	0.025	0.024	0.012

Se comprobó que para obtener una mayor precisión de hasta 6 decimales es necesario implementar al menos 20 iteraciones, así como incrementar el ancho de la palabra en la arquitectura por lo menos a 24 bits, de esta manera se mejoraría la precisión considerablemente.

En la tabla 4 se muestran los tiempos de ejecución del algoritmo en ciclos de reloj. Se puede apreciar que para las funciones de seno y coseno solo se requieren 16 ciclos de reloj, mientras que para las funciones de tangente, secante y cosecante se incrementan hasta 36 ciclos.

Tabla 4 Tiempo de ejecución de TLU.

Ciclos de reloj por función	
Función	**Ciclos de reloj**
seno	16
coseno	16
tangente	35
cosecante	36
secante	36

Se preparó el diseño físico con dimensiones de 1.88 mm², el cual se muestra en la figura 7 junto al diagrama de conexiones para un encapsulado DIP40. Una vez obtenido el diseño, se volvió a realizar la verificación de reglas de diseño y

colocación de pads. El diseño de la unidad lógica trigonométrica se envió a fabricar a través del programa educativo con MOSIS.

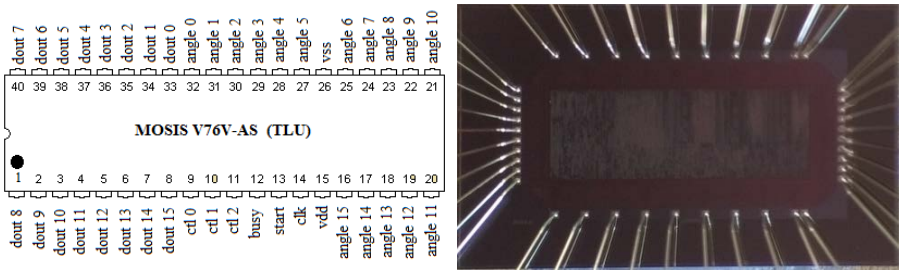

Figura 7 Configuración de las terminales del encapsulado y foto del chip TLU.

4. Discusión

En el presente trabajo se muestran los resultados de una arquitectura VLSI que resuelve funciones trigonométricas utilizando el algoritmo CORDIC mediante métodos iterativos. Esta arquitectura presenta errores relativos de ±0.034% debido al truncamiento de datos y el número de iteraciones implementadas. Sin embargo, los resultados de simulación en chip mostraron una eficiente implementación del algoritmo CORDIC bajo una arquitectura simplificada a 16 bits que se realizó con la finalidad de obtener nuestros propios módulos IP (propiedad intelectual), indispensables en el diseño de circuitos integrados para aplicaciones que requieren un alto grado de eficiencia. Para nuestro grupo de investigación es de particular importancia la generación de nuestros propios diseños a nivel VLSI, lo que nos permitirá ir creciendo y madurando los diseños a niveles más complejos y a la vez poder ser implementados en aplicaciones que requieran un alto desempeño matemático.

5. Bibliografía y Referencias

[1] Bruce H. Edwards, Como multiplican y dividen las calculadoras, Vol VIII. 2000.

[2] Chaput J-P., Pétrot F. GenLib User's Manual [en línea], Pierre & Marie Curie University, LIP6 ASIM Department, 2002: //ftp.lip6.fr/lip6/softs/alliance/latest-checkout/alliance/src/genlib/doc/genlib.pdf.

[3] Chen, Chuen-Yau y Wen-Chih Liu. Architecture for CORDIC algorithm realization without ROM lookup tables. 2003.p. IV-544-IV-547 vol.4.

[4] Duprat, J. y Muller, J. The CORDIC algorithm: new results for fast VLSI implementation. IEEE Transactions on Computers. 1993, vol 42, núm. 2, p. 168-178.

[5] Lam K-S. Ak F. Alliance Tutorial, Part 1 VHDL Modeling and Simulation [en línea], Pierre & Marie Curie University, France, 2004. Disponible en: https://www-soc.lip6.fr/equipe-cian/logiciels/alliance/

[6] LIP6, Alliance VLSI CAD System, [en línea], Pierre & Marie Curie University, Paris, France, 2018: https://www-soc.lip6.fr/equipe-cian/logiciels/alliance/.

[7] Muller, J.-M. (2016). Elementary Functions: Algorithms and Implementation. New York: Springer Science+Business Media.

[8] Pramod K. Meher, Javier Valls. 50 Years of CORDIC Algorithms, Architectures and Applications. IEEE Transactions on Circuits and Systems. 2009.

[9] Takagi, N., Asada, T. y Yajima, S. . Redundant CORDIC methods with a constant scale factor for sine and cosine computation. IEEE Transactions on Computers. 1991, vol 40, núm. 9, p. 989-995.

[10] The MOSIS Service, USC Information Sciences Institute, 2018, disponible en: https://www.mosis.com/what-is-mosis.

[11] Valls, Javier. Evaluation of CORDIC Algorithms for FPGA Design. Journal of VLSI Signal Processing. 2000.

[12] Volder, Jack. The CORDIC Computing Technique. Proceedings of the Western Joint Computer Conference. 1959.

[13] Walter, J.S. A unified algorithm for elementary functions. Spring Joint Computer Conference. 1971.

[14] Walter, J.S. The Story of Unified CORDIC. Journal of VLSI Signal Processing. 2000.

[15] Weste N-H. E., Harris D-M., CMOS VLSI Design a Circuit and Systems Perspective, 4th ed., USA, Addison Wesley, 2011, pp. 615-657.